Pogrammation Impérative

PULUDISU Mpia Mimpiya
Numéro étudiant : 18913467
L1 Informatique

26 août 2024

Table des matières

1 Attendus

Pour ce cours, nous devons rendre les exercices cx17.4, cx17.7 et cx17.8 de la série cx17 et les exercices cx25.0 et cx25.1 de la série cx25. Comme certains exercices dépendent d'autres exercices dans le cours, j'ai ajouté exercice cx15.3 de la série cx15 et les exercices cx18.1 et cx18.6 de la série cx18.

2 Série cx18

2.1 cx18.1 : Recoder la fonction putlist() pour afficher la liste en ordre inverse

```
int main(int k) // k utlisée comme locale
{ list L = nil ;
    for (k = 'a' ; k < 'i' ; k++)
        L = cons(k, L) ;
    putlist(L) ;
    return 0 ; }

list cons(int car, list L)
{ list new = malloc(sizeof(node)) ;
    if (! new) usage("cons : manque de RAM") ; // enfin, un peu de sérieux !
    new -> car = car ;
    new -> cdr = L ;
    return new ; }

void putlist(list L)
    { if (! L) return ; // nil : fin de liste
        printf("%c ", L -> car) ;
    putlist(L -> cdr) ; }
```

Dans le programme ci-dessus, recoder la fonction putlist() pour qu'elle affiche la liste en ordre inverse, i.e. l'ordre dans lequel les éléments ont été rencontrés et empilés.

Pour afficher la liste en ordre inverse, nous devons inverser la liste avant de l'afficher, voici un petit programme qui réalise cette tâche :

```
/* ********************************************************
 * Nom            : cx18.1.c
 * Rôle           : Affiche la liste en ordre inverse
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -W cx18.1.c -o main
 * Usage          : ./main
 * ********************************************************/

#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int car;
    struct node *cdr;
} *list;

list cons(int car, list L) {
    list new = malloc(sizeof(struct node));
    if (!new) {
        fprintf(stderr, "cons : manque de RAM\n");
        exit(1);
    }
    new->car = car;
    new->cdr = L;
    return new;
}
```

```c
void putlist(list L) {
    if (!L) return; // nil : fin de liste
    printf("%c ", L->car);
    putlist(L->cdr);
}

int main(int k) {
    list L = NULL;
    for (k = 'a'; k < 'i'; k++) {
        L = cons(k, L);
    }
    putlist(L);
    return 0;
}
```

Nous pouvons compiler et lancer le programme :

```
gcc -W cx18.1.c -o cx18.1
./cx18.1
```

2.2 cx18.6 : Déporter les fonctions de gestion de listes dans une bibliothèque dynamique

Adapter votre solution de [cx18.1] en déportant les fonctions de gestion de listes dans une bibliothèque dynamique vérifier que le programme tourne toujours.
Essayer d'adapter de la même façon les autres exemples de la série...

Pour déporter les fonctions de gestion de listes dans une bibliothèque dynamique, nous allons créer les fichiers suivants dans la même arborecence :

1. `list.h` : pour les déclarations. Voici le code de ce fichier :

```c
#ifndef LIST_H
#define LIST_H

#include <stdlib.h>

typedef struct node {
    int car;
    struct node *cdr;
} *list;

list cons(int car, list L);
void putlist(list L);
list reverse_list(list L);

#endif
```

2. `list.c` : pour les définitions. Voici le code de ce fichier :

```c
#include <stdio.h>
#include "list.h"

list cons(int car, list L) {
    list new = malloc(sizeof(struct node));
    if (!new) {
        fprintf(stderr, "cons : manque de RAM\n");
        exit(1);
    }
    new->car = car;
    new->cdr = L;
    return new;
}

void putlist(list L) {
    L = reverse_list(L);
    while (L != NULL) {
        printf("%c ", L->car);
        L = L->cdr;
    }
    printf("\n");
}

list reverse_list(list L) {
    list prev = NULL;
    list current = L;
    list next = NULL;

    while (current != NULL) {
        next = current->cdr;
        current->cdr = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

3. Et `main.c` : pour le programme principal. Voici le code de ce fichier :

```c
/* ********************************************************
 * Nom            : main.c
 * Rôle           : Déportation de fonctions de gestion de listes
 *                  dans une bibliothèque dynamique
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -c -fPIC list.c -o list.o
 *                  gcc -shared -o list.so list.o
 *                  gcc -o main main.c -L. -llist
 * Usage          : ./main
 * ********************************************************/
#include "list.h"

int main(int argc, char *argv[]) {
    list L = NULL;
    int k;
```

```
    for (k = 'h'; k >= 'a'; k--) {
        L = cons(k, L);
    }
    putlist(L);
    return 0;
}
```

Pour compiler et exécuter le programme,voici les étapes à suivres :

1. Compilation et création de la bibliothèque dynamique :
   - Compilez le fichier `list.c` pour créer un objet partagé :

     ```
     gcc -c -fPIC list.c -o list.o
     ```

   - Créez la bibliothèque dynamique à partir de l'objet partagé :

     ```
     gcc -shared -o liblist.so list.o
     ```

   - Compilez le programme principal en liant la bibliothèque dynamique :

     ```
     gcc -o main main.c -L. -llist
     ```

2. Exécuter le programme : Pour exécuter le programme, nous devons nous assurer que le système peut trouver la bibliothèque dynamique. Nous définissons la variable d'environnement `LD_LIBRARY_PATH` pour inclure le répertoire courant :

   ```
   export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
   ./main
   ```

Si toutes les étapes sont respectées, on devait avoir `h g f e d c b a`.

3 Série cx15

3.1 cx15.3 : Créer le fichier texte `cx15.3.data` et lire les données dans une table de chaînes

Voici le contenu du fichier `cx15.3.data` :

```
Joe Jack William Averell
```

notre petit programme :

```c
/* ********************************************************
 * Nom            : main.c
 * Rôle           : Lis les informations dans un fichier
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -Wall main.c -o main
 * Usage          : ./main
 * ********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAMES 100
#define MAX_NAME_LENGTH 50

int main() {
    FILE *file = fopen("cx15.3.data", "r");
    if (!file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier\n");
        return 1;
    }

    char names[MAX_NAMES][MAX_NAME_LENGTH];
    int count = 0;

    while (fscanf(file, "%s", names[count]) != EOF) {
        count++;
    }

    fclose(file);

    for (int i = 0; i < count; i++) {
        printf("%s\n", names[i]);
    }

    return 0;
}
```

La compilation et l'exécution du programme :

```
gcc -W main.c -o main
./main
```

La sortie du programme est :

```
Joe
Jack
William
Averell
```

4 Série cx17

4.1 cx17.2 : Utiliser la solution de [cx15.3] pour lire et exploiter une STOPLIST

Utiliser votre solution de [cx15.3] pour rajouter au programme la capacité de lire et d'exploiter une STOPLIST comme dans le prototype PYTHON.

Pour cette étape, nous allons lire une STOPLIST à partir d'un fichier et l'utiliser pour filtrer les mots lus dans cx15.3.data.

Voici les fichiers de notre arborecence :

```
~/Bureau/IMPERATIVE/codes/cx17/cx17.2 (0.083s)
tree

.
├── cx15.3.data
├── main.c
└── stoplist.txt

0 directories, 3 files
```

Voici le programme principal :

```c
/* ********************************************************
 * Nom            : main.c
 * Rôle           : Lis les mots dans un fichier et exclue les
 *                  mots figurant la stoplist
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -Wall main.c -o main
 * Usage          : ./main
 * ********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAMES 100
#define MAX_NAME_LENGTH 50
#define MAX_STOPLIST 100

int is_in_stoplist(char *word, char stoplist[][MAX_NAME_LENGTH], int
    stoplist_count) {
    for (int i = 0; i < stoplist_count; i++) {
        if (strcmp(word, stoplist[i]) == 0) {
            return 1;
        }
    }
    return 0;
}

int main() {
    FILE *file = fopen("cx15.3.data", "r");
    if (!file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier\n");
        return 1;
    }

    char names[MAX_NAMES][MAX_NAME_LENGTH];
    int count = 0;
```

```
    while (fscanf(file, "%s", names[count]) != EOF) {
        count++;
    }

    fclose(file);

    FILE *stoplist_file = fopen("stoplist.txt", "r");
    if (!stoplist_file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier stoplist\n");
        return 1;
    }

    char stoplist[MAX_STOPLIST][MAX_NAME_LENGTH];
    int stoplist_count = 0;

    while (fscanf(stoplist_file, "%s", stoplist[stoplist_count]) != EOF) {
        stoplist_count++;
    }

    fclose(stoplist_file);

    for (int i = 0; i < count; i++) {
        if (!is_in_stoplist(names[i], stoplist, stoplist_count)) {
            printf("%s\n", names[i]);
        }
    }

    return 0;
}
```

La compilation et l'exécution du programme :

```
gcc -W main.c -o main
./main
```

La sortie du programme est :

```
William
Averell
```

4.2 cx17.6 : Utiliser une STOPLIST représentée en interne sous la forme d'une liste élastique

Utiliser votre solution de [cx15.3] et de [cx17.2] pour ajouter au programme la capacité d'accueillir et d'exploiter une STOPLIST représentée en interne sous la forme d"une liste élastique, et lue à partir d'un fichier dont le nom sera spécifié sur la LdC s'il ne l'est pas, le programme ouvrira un fichier par défaut.

Pour cet exercice, nous allons utiliser une liste chaînée pour représenter la STOPLIST.

Pour réaliser cet exercice, voici les fichiers de notre arborecence :

```
~/Bureau/IMPERATIVE/codes/cx17/cx17.7 (0.08s)
tree

.
├── cx15.3.data
├── list.c
├── list.h
├── main.c
└── stoplist.txt

0 directories, 5 files
```

Nous allons nous attarder sur les trois fichiers principaux :

1. `list.h` :

```c
#ifndef LIST_H
#define LIST_H

typedef struct node {
    char *car;
    struct node *cdr;
} *list;

list cons(char *car, list L);
void putlist(list L);
int is_in_list(list L, char *word);

#endif
```

2. `list.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

list cons(char *car, list L) {
    list new = malloc(sizeof(struct node));
    if (!new) {
        fprintf(stderr, "cons : manque de RAM\n");
        exit(1);
    }
    new->car = strdup(car);
    new->cdr = L;
    return new;
}

void putlist(list L) {
    if (!L) return; // nil : fin de liste
    putlist(L->cdr);
```

```
    printf("%s ", L->car);
}

int is_in_list(list L, char *word) {
    while (L) {
        if (strcmp(L->car, word) == 0) {
            return 1;
        }
        L = L->cdr;
    }
    return 0;
}
```

3. main.c :

```
/* ******************************************************
 * Nom            : main.c
 * Rôle           : Lis les mots dans un fichier et exclue les
 *                  mots figurant la stoplist
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ******************************************************
 * Compilation    : gcc -c list.c -o list.o
                    gcc -c main.c -o main.o
                    gcc main.o list.o -o main
 * Usage          : ./main
 * ******************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

#define MAX_NAMES 100
#define MAX_NAME_LENGTH 50

int main(int argc, char *argv[]) {
    FILE *file = fopen("cx15.3.data", "r");
    if (!file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier\n");
        return 1;
    }

    char names[MAX_NAMES][MAX_NAME_LENGTH];
    int count = 0;

    while (fscanf(file, "%s", names[count]) != EOF) {
        count++;
    }

    fclose(file);

    const char *stoplist_filename = (argc > 1) ? argv[1] : "stoplist.
       txt";
    FILE *stoplist_file = fopen(stoplist_filename, "r");
    if (!stoplist_file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier stoplist
           \n");
        return 1;
```

```
    }

    list stoplist = NULL;
    char word[MAX_NAME_LENGTH];

    while (fscanf(stoplist_file, "%s", word) != EOF) {
        stoplist = cons(word, stoplist);
    }

    fclose(stoplist_file);

    for (int i = 0; i < count; i++) {
        if (!is_in_list(stoplist, names[i])) {
            printf("%s\n", names[i]);
        }
    }

    return 0;
}
```

La compilation et l'exécution du programme :

```
gcc -c list.c -o list.o
gcc -c main.c -o main.o
gcc main.o list.o -o main
./main
```

La sortie du programme, on a le résultat suivant en prenant compte de notre `cx15.3.data` et de notre `stoplist.txt` :

```
William
Averell
```

4.3 cx17.7 : Adapter la solution de [cx17.6] pour utiliser la bibliothèque créée pour [cx18.6]

Adapter votre solution de [cx17.6] pour utiliser la bibliothèque créée pour [cx18.6].

Pour cette étape, nous allons utiliser la bibliothèque créée pour [cx18.6].

Cet exercice est presque similaire à [cx17.6], nous allons donc aller rapidement :

Voici les codes de fichiers principaux :

1. `list.h` :

```c
#ifndef LIST_H
#define LIST_H

typedef struct node {
    char *car;
    struct node *cdr;
} *list;

list cons(char *car, list L);
void putlist(list L);
int is_in_list(list L, char *word);

#endif
```

2. `list.c` :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

list cons(char *car, list L) {
    list new = malloc(sizeof(struct node));
    if (!new) {
        fprintf(stderr, "cons : manque de RAM\n");
        exit(1);
    }
    new->car = strdup(car);
    new->cdr = L;
    return new;
}

void putlist(list L) {
    if (!L) return; // nil : fin de liste
    putlist(L->cdr);
    printf("%s ", L->car);
}

int is_in_list(list L, char *word) {
    while (L) {
        if (strcmp(L->car, word) == 0) {
            return 1;
        }
        L = L->cdr;
    }
    return 0;
}
```

3. main.c :

```c
/* ********************************************************
 * Nom            : main.c
 * Rôle           : Lis les mots dans un fichier et exclue les
 *                  mots figurant la stoplist (adaptation de cx15.3)
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -c -fPIC list.c -o list.o
                    gcc -shared -o list.so list.o
                    gcc -o main main.c -L. -llist
 * Usage          : export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
 *                  ./main
 * ********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

#define MAX_NAMES 100
#define MAX_NAME_LENGTH 50

int main(int argc, char *argv[]) {
    FILE *file = fopen("cx15.3.data", "r");
    if (!file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier\n");
        return 1;
    }

    char names[MAX_NAMES][MAX_NAME_LENGTH];
    int count = 0;

    while (fscanf(file, "%s", names[count]) != EOF) {
        count++;
    }

    fclose(file);

    const char *stoplist_filename = (argc > 1) ? argv[1] : "stoplist.
       txt";
    FILE *stoplist_file = fopen(stoplist_filename, "r");
    if (!stoplist_file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier stoplist
           \n");
        return 1;
    }

    list stoplist = NULL;
    char word[MAX_NAME_LENGTH];

    while (fscanf(stoplist_file, "%s", word) != EOF) {
        stoplist = cons(word, stoplist);
    }

    fclose(stoplist_file);

    for (int i = 0; i < count; i++) {
        if (!is_in_list(stoplist, names[i])) {
```

```
            printf("%s\n", names[i]);
        }
    }

    return 0;
}
```

La compilation et l'exécution du programme, voici les étapes à suivre :

1. **Compilation et création de la bibliothèque dynamique**
   (a) Compilez le fichier `list.c` pour créer un objet partagé :

```
gcc -c -fPIC list.c -o list.o
```

   (b) Créez la bibliothèque dynamique à partir de l'objet partagé :

```
gcc -shared -o list.so list.o
```

   (c) Compilez le programme principal en liant la bibliothèque dynamique :

```
gcc -o main main.c -L. -llist
```

2. **Exécution du programme**

```
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
./main
```

4.4 cx17.8 : Adapter la solution de [cx17.6] pour utiliser la bibliothèque créée pour [cx18.7]

1. `list.h` :

```
#ifndef LIST_H
#define LIST_H

typedef struct node {
    char *car;
    struct node *cdr;
} *list;

list cons(char *car, list L);
void putlist(list L);
int is_in_list(list L, char *word);

#endif
```

2. `list.c` :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

list cons(char *car, list L) {
    list new = malloc(sizeof(struct node));
```

```
    if (!new) {
        fprintf(stderr, "cons : manque de RAM\n");
        exit(1);
    }
    new->car = strdup(car);
    new->cdr = L;
    return new;
}

void putlist(list L) {
    if (!L) return; // nil : fin de liste
    putlist(L->cdr);
    printf("%s ", L->car);
}

int is_in_list(list L, char *word) {
    while (L) {
        if (strcmp(L->car, word) == 0) {
            return 1;
        }
        L = L->cdr;
    }
    return 0;
}
```

3. main.c :

```
/* ********************************************************
 * Nom            : main.c
 * Rôle           : Lis les mots dans un fichier et exclue les
 *                  mots figurant la stoplist (adaptation de cx17.6)
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -c -fPIC list.c -o list.o
                    gcc -shared -o list.so list.o
                    gcc -o main main.c -L. -llist
 * Usage          : export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
 *                  ./main
 * ********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"

#define MAX_NAMES 100
#define MAX_NAME_LENGTH 50

int main(int argc, char *argv[]) {
    FILE *file = fopen("cx15.3.data", "r");
    if (!file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier\n");
        return 1;
    }

    char names[MAX_NAMES][MAX_NAME_LENGTH];
    int count = 0;
```

```
    while (fscanf(file, "%s", names[count]) != EOF) {
        count++;
    }

    fclose(file);

    const char *stoplist_filename = (argc > 1) ? argv[1] : "stoplist.
        txt";
    FILE *stoplist_file = fopen(stoplist_filename, "r");
    if (!stoplist_file) {
        fprintf(stderr, "Erreur lors de l'ouverture du fichier stoplist
            \n");
        return 1;
    }

    list stoplist = NULL;
    char word[MAX_NAME_LENGTH];

    while (fscanf(stoplist_file, "%s", word) != EOF) {
        stoplist = cons(word, stoplist);
    }

    fclose(stoplist_file);

    for (int i = 0; i < count; i++) {
        if (!is_in_list(stoplist, names[i])) {
            printf("%s\n", names[i]);
        }
    }

    return 0;
}
```

Pour compiler et exécuter le programme, voir l'exercice cx17.7 c'est quasi le même processus.

5 Série cx25

5.1 cx25.0 : Programmer cet émulateur de sorte qu'il puisse lire le code numérique (ou symbolique) à partir d'un fichier

Ce petit programme écrit en C simule un processeur simple avec une mémoire de taille fixe et un ensemble d'instructions de base. Il charge un programme à partir d'un fichier (program.hex), l'exécute et affiche l'état de l'accumulateur (A) et du compteur de programme (PC) à chaque étape.

```c
/* ********************************************************
 * Nom            : cx25.0.c
 * Rôle           : Emulateur d'un programme de lecture numérique
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ********************************************************
 * Compilation    : gcc -Wall cx25.0.c -o main
 * Usage          : ./main program.hex
 * ********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MEMORY_SIZE 256
#define PROGRAM_START 0x50  // Adresse de début du programme

unsigned char memory[MEMORY_SIZE];
unsigned char A = 0;  // Accumulateur
// Compteur de programme, commence à l'adresse de début
unsigned int PC = PROGRAM_START;

void charger_programme(const char *nom_fichier) {
    FILE *file = fopen(nom_fichier, "r");
    if (!file) {
        perror("Échec de l'ouverture du fichier");
        exit(EXIT_FAILURE);
    }

    int adresse;
    unsigned char valeur1, valeur2;
    while (fscanf(file, "%x %hhx %hhx", &adresse, &valeur1, &valeur2) == 3)
       {
        if (adresse >= MEMORY_SIZE - 1) {
            fprintf(stderr, "Adresse invalide : %x\n", adresse);
            fclose(file);
            exit(EXIT_FAILURE);
        }
        memory[adresse] = valeur1;
        memory[adresse + 1] = valeur2;
    }

    fclose(file);
}

void executer() {
    while (PC < MEMORY_SIZE) {
        unsigned char opcode = memory[PC++];
        unsigned char operand;
```

```
        switch (opcode) {
            case 0x00: printf("Instruction HALT rencontrée. Arrêt de l'exé
                cution.\n"); return;
            case 0x40: A += memory[PC++]; break;
            case 0x41: A -= memory[PC++]; break;
            case 0x48: A = memory[PC++]; break;
            case 0x49: A &= memory[PC++]; break;
            case 0x4A: A |= memory[PC++]; break;
            case 0x4B: A ^= memory[PC++]; break;
            case 0x4C: A = ~A; break;
            case 0x4D: A <<= 1; break;
            case 0x4E: A >>= 1; break;
            case 0x4F: A += 1; break;
            case 0x50: A -= 1; break;
            case 0x51: A *= 2; break;
            case 0x52: A /= 2; break;
            case 0x53: A &= 0xFF; break;
            case 0x54: A |= 0xFF; break;
            default: printf("Opcode inconnu : %02x\n", opcode); return;
        }

        printf("PC : %02x, A après exécution : %02x\n", PC - 1, A);
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Utilisation : %s <fichier programme>\n", argv[0]);
        return EXIT_FAILURE;
    }

    memset(memory, 0, MEMORY_SIZE);
    charger_programme(argv[1]);

    // Débogage : Afficher le contenu de la mémoire
    printf("Contenu de la mémoire avant exécution :\n");
    for (int i = 0; i < MEMORY_SIZE; i++) {
        printf("%02x ", memory[i]);
        if ((i + 1) % 16 == 0) printf("\n");
    }
    printf("\n");

    executer();
    printf("Valeur finale de A : %02x\n", A);

    return EXIT_SUCCESS;
}
```

Pour compiler et exécuter le programme, merci de faire :

```
gcc -W cx25.0.c -o main
./main program.hex
```

```
~/Bureau/IMPERATIVE/codes/cx25/cx25.0 (0.107s)
gcc -W cx25.0.c -o main
```

```
~/Bureau/IMPERATIVE/codes/cx25/cx25.0 (0.089s)
./main program.hex
Contenu de la mémoire avant exécution :
00 76 00 00 00 00 00 00 00 00 00 00 00 00 00 00
78 88 88 7c 00 00 00 00 00 00 00 00 00 00 00 00
00 01 7e 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
73 8a 00 00 00 00 00 00 71 8c 00 00 00 00 00 00
49 70 40 70 48 71 48 72 00 5e 48 8d 10 74 40 71
72 84 40 70 48 71 00 6c 48 8d 10 74 41 71 10 6e
00 71 00 00 74 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

PC : 51, A après exécution : 00
PC : 53, A après exécution : 70
PC : 55, A après exécution : 71
PC : 57, A après exécution : 72
Instruction HALT rencontrée. Arrêt de l'exécution.
Valeur finale de A : 72
```

5.2 cx25.1 : Coder la boucle d'évaluation pour en faire un STEPPER

Coder la boucle d'évaluation pour en faire un STEPPER qui affiche la valeur de PC et de A, et attend confirmation au clavier pour exécuter l'instruction suivante !; développez-le pour émuler celles des fonctionnalités de gdb qui vous paraissent utiles ou faciles.

Ce petit programme écrit en C simule un processeur simple avec une mémoire de taille fixe et un ensemble d'instructions de base. Il charge un programme à partir d'un fichier (program.hex), l'exécute et affiche l'état de l'accumulateur (A) et du compteur de programme (PC) à chaque étape. A la différence de cx25.0, le programme attend confirmation au clavier pour exécuter l'instruction suivante.

Voici le code source :

```c
/* ******************************************************
 * Nom            : cx25.1.c
 * Rôle           : Emulateur d'un programme de lecture numérique
 *                  Utilisant une boucle pour faire un stepper
 * Auteur         : Mpia Mimpiya PULUDISU
 * Version        : 1.0
 * Date           : 2024-08-26
 * Licence        : L1 PROGC
 * ******************************************************
 * Compilation    : gcc -Wall cx25.1.c -o main
 * Usage          : ./main program.hex
 * ******************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MEMORY_SIZE 256
#define PROGRAM_START 0x50  // Adresse de début du programme

unsigned char memory[MEMORY_SIZE];
unsigned char A = 0;  // Accumulateur
// Compteur de programme, commence à l'adresse de début
unsigned int PC = PROGRAM_START;

void charger_programme(const char *nom_fichier) {
    FILE *file = fopen(nom_fichier, "r");
    if (!file) {
        perror("Échec de l'ouverture du fichier");
        exit(EXIT_FAILURE);
    }

    int adresse;
    unsigned char valeur1, valeur2;
    while (fscanf(file, "%x %hhx %hhx", &adresse, &valeur1, &valeur2) == 3)
       {
        if (adresse >= MEMORY_SIZE - 1) {
            fprintf(stderr, "Adresse invalide : %x\n", adresse);
            fclose(file);
            exit(EXIT_FAILURE);
        }
        memory[adresse] = valeur1;
        memory[adresse + 1] = valeur2;
    }

    fclose(file);
}

void executer() {
    while (PC < MEMORY_SIZE) {
        unsigned char opcode = memory[PC++];
        unsigned char operand;

        printf("PC : %02x, A avant exécution : %02x\n", PC - 1, A);
        printf("Appuyez sur Entrée pour continuer...\n");
        getchar();  // Attendre une entrée de l'utilisateur

        switch (opcode) {
            case 0x00: printf("Instruction HALT rencontrée. Arrêt de l'exé
               cution.\n"); return;
```

```
            case 0x40: A += memory[PC++]; break;
            case 0x41: A -= memory[PC++]; break;
            case 0x48: A = memory[PC++]; break;
            case 0x49: A &= memory[PC++]; break;
            case 0x4A: A |= memory[PC++]; break;
            case 0x4B: A ^= memory[PC++]; break;
            case 0x4C: A = ~A; break;
            case 0x4D: A <<= 1; break;
            case 0x4E: A >>= 1; break;
            case 0x4F: A += 1; break;
            case 0x50: A -= 1; break;
            case 0x51: A *= 2; break;
            case 0x52: A /= 2; break;
            case 0x53: A &= 0xFF; break;
            case 0x54: A |= 0xFF; break;
            default: printf("Opcode inconnu : %02x\n", opcode); return;
        }

        printf("PC : %02x, A après exécution : %02x\n", PC - 1, A);
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Utilisation : %s <fichier programme>\n", argv[0]);
        return EXIT_FAILURE;
    }

    memset(memory, 0, MEMORY_SIZE);
    charger_programme(argv[1]);

    // Débogage : Afficher le contenu de la mémoire
    printf("Contenu de la mémoire avant exécution :\n");
    for (int i = 0; i < MEMORY_SIZE; i++) {
        printf("%02x ", memory[i]);
        if ((i + 1) % 16 == 0) printf("\n");
    }
    printf("\n");

    executer();
    printf("Valeur finale de A : %02x\n", A);

    return EXIT_SUCCESS;
}
```

Il ne reste plus qu'à compiler et exécuter notre programme :

```
gcc -W cx25.1.c -o main
./main program.hex
```

```
~/Bureau/IMPERATIVE/codes/cx25/cx25.1 (0.134s)
gcc -W cx25.1.c -o main
```

```
~/Bureau/IMPERATIVE/codes/cx25/cx25.1 (24.808s)
./main program.hex
Contenu de la mémoire avant exécution :
00 76 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 88 7c 00 00 00 00 00 00 00 00 00 00 00 00
00 01 7e 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
71 7a 00 00 00 00 00 00 71 80 00 00 00 00 00 00
49 70 40 70 48 71 48 72 00 5e 48 8d 10 74 40 71
48 72 40 70 48 71 00 6c 48 8d 10 74 41 71 10 6e
00 71 00 00 74 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

PC : 50, A avant exécution : 00
Appuyez sur Entrée pour continuer...

PC : 51, A après exécution : 00
PC : 52, A avant exécution : 00
Appuyez sur Entrée pour continuer...

PC : 53, A après exécution : 70
PC : 54, A avant exécution : 70
Appuyez sur Entrée pour continuer...

PC : 55, A après exécution : 71
PC : 56, A avant exécution : 71
Appuyez sur Entrée pour continuer...

PC : 57, A après exécution : 72
PC : 58, A avant exécution : 72
Appuyez sur Entrée pour continuer...

Instruction HALT rencontrée. Arrêt de l'exécution.
Valeur finale de A : 72
```